Getting a Quick Performance Overview with Intel APS

The Intel Application Performance Snapshot (APS) tool provides a quick overview of your application's performance on processor and memory usage, message passing interface (MPI), and I/O, as well as load imbalance between threads or processes. In addition to the performance metrics listed below, APS also provides suggestions for performance enhancement opportunities and additional Intel profiling tools you can use to get more in-depth analysis.

Note: Although the Intel APS tool can be used for any of the Pleiades, Aitken, and Electra Intel Xeon processor types, it does not fully support the Sandy Bridge and Haswell processor types. APS cannot be used for the Aitken Rome processors.

Before You Begin

Note the following information:

- The Intel C/C++ or Fortran Compiler is not required, but is recommended. However, analysis of OpenMP imbalance is only available for applications that use the Intel OpenMP runtime library.
- There is no support for OpenMPI.
- Analysis of MPI imbalance is only available when you are using the Intel MPI Library version 2017 and later. (With the exception of the MPI imbalance metric, APS MPI analysis works for applications that use the NAS-recommended HPE MPT library.)

TIP: The NAS-developed MPIProf tool provides similar MPI analysis with MPI imbalance information included.

Running an APS Analysis

On Pleiades, APS is available via the **vtune/2021.9** module. See **aps --help** for available APS options.

To run an analysis for serial and OpenMP applications:

```
module load vtune/2021.9
aps <options> ./a.out
```

To run an analysis for MPI applications:

```
module load vtune/2021.9

module load mpi-hpe/mpt.2.25

setenv MPI_SHEPHERD true
setenv MPI_USING_VTUNE true
mpiexec -np xx aps <options> ./a.out
```

TIP: The process/thread pinning tools **mbind.x** and **omplace** can be used with **aps**. Either of the following two methods may work:

- **mpiexec -np xx aps mbind (or omplace) <mbind or omplace *options*> a.out**
- **mpiexec -np xx mbind (or omplace) <mbind or omplace *options*> aps <*options*> a.out**

At the end of the run, APS provides a directory that contains the data collected during the analysis. The default name of the directory is **aps_result_*yyyymmdd***.

WARNING: The directory will be overwritten if you run another APS analysis in the same directory on the same day.

To send the results to another directory, you can include **--result-dir=*dir_name*** or **-r=*dir_name*** option in the **aps** command line (where **dir_name** represents the directory name of your choice).

Viewing the Report

APS provides a text summary and an HTML file named **aps_report*_yyyymmdd_hhmmss*.html** (where **yyyymmdd_hhmmss** is the date and time when the HTML file is created). The HTML file contains the same information as the text summary, but also offers the following features when you open the file with a web browser:

- A description of each metric is shown when you hover your mouse over the metric name.
- Possible performance issues of your run are highlighted in red.
- Suggestions with links to Intel tools are provided to help with performance enhancement.

Note: NAS security policy prohibits using web browsers on Pleiades. To view the HTML file, transfer it to your own workstation.

For Serial and OpenMP Applications

For serial or OpenMP applications, no action is required to generate the reports. At the end of the run, APS automatically generates the text summary (appended to your application's **stdout**) and the HTML file.

For MPI Applications

For MPI applications, you must generate the text summary (shown on your screen) and HTML file after the analysis completes, as follows:

```
aps --report=dir_name
```

where **dir_name** is the name of the directory created at the end of the analysis run.

If the summary report shows that your application is MPI-bound, run the following command to get more details about message passing, such as message sizes, data transfers between ranks or nodes, and time in collective operations:

```
aps-report <options> dir_name
```

See **aps-report --help** for available options.

You can also rerun the analysis after setting additional environmental variables. For example:

```
setenv MPS_STAT_LEVEL n
or
setenv APS_STAT_LEVEL n
```

where *n* is 2, 3 or 4 to get more detailed information on your application's MPI performance.

Note: If MPS_STAT_LEVEL (or APS_STAT_LEVEL) is set to 3 or 4, you must use mpi-hpe/mpt.2.18r160 or a later version when you run the analysis to avoid a segmentation fault (SEGV).

Quick Metrics Reference

Intel APS collects the metrics described in the following list.

Note: These descriptions are adapted from the Application Performance Snapshot User's Guide for Linux OS, where you can find additional details about each metric.

Elapsed Time
: Execution time of specified application in seconds.

SP GFLOPS
: Number of single precision giga-floating point operations (gigaflops) calculated per second. SP GFLOPS metrics are only available for 3rd Generation Intel Core processors, 5th Generation Intel processors, and 6th Generation Intel processors.

DP GFLOPS
: Number of double precision giga-floating point operations calculated per second. DP GFLOPS metrics are only available for 3rd Generation Intel Core processors, 5th Generation Intel processors, and 6th Generation Intel processors.

Cycles per Instruction Retired (CPI) Rate
: The amount of time each executed instruction took measured by cycles. A CPI of 1 is considered acceptable for high performance computing (HPC) applications, but different application domains will have varied expected values. The CPI value tends to be greater when there is long-latency memory, floating-point, or SIMD operations, non-retired instructions due to branch mis-predictions, or instruction starvation at the front end.

CPU Utilization
: Helps evaluate the parallel efficiency of your application. Estimates the utilization of all the logical CPU cores in the system by your application. 100% utilization means that your application keeps all the logical CPU cores busy for the entire time that it runs. Note that the metric does not distinguish between useful application work and the time that is spent in parallel runtimes.

MPI Time
: Time spent inside the MPI library. Values more than 15% might need additional exploration on MPI communication efficiency. This might be caused by high wait times inside the library, active communications, non-optimal settings of the MPI library. See MPI Imbalance metric to see if the application has load balancing problem.

MPI Imbalance
: Mean unproductive wait time per process spent in the MPI library calls when a process is waiting for data.

Serial Time
: Time spent by the application outside any OpenMP region in the master thread during collection. This directly impacts application Collection Time and scaling. High values might signal a performance problem to be solved via code parallelization or algorithm tuning.

OpenMP Imbalance
: Indicates the percentage of elapsed time that your application wastes at OpenMP synchronization barriers because of load imbalance.

Memory Stalls
: Indicates how memory subsystem issues affect the performance. Measures a fraction of slots where pipeline could be stalled due to demand load or store instructions. This metric's value can indicate that a significant fraction of execution pipeline slots could be stalled due to demand memory load and stores. See the second level metrics to define if the application is cache- or DRAM-bound and the NUMA efficiency.

Cache Stalls
: indicates how often the machine was stalled on L1, L2, and L3 cache. While cache hits are serviced much more quickly than hits in DRAM, they can still incur a significant performance penalty. This metric also includes coherence penalties for shared data.

DRAM Stalls
: Indicates how often the CPU was stalled on the main memory (DRAM) because of demand loads or stores.

DRAM Bandwidth
: The metrics in this section indicate the extent of high DRAM bandwidth utilization by the system during elapsed time. They include:
  - **Average Bandwidth:** Average memory bandwidth used by the system during elapsed time.
  - **Peak:** Maximum memory bandwidth used by the system during elapsed time.
  - **Bound:** The portion of elapsed time during which the utilization of memory bandwidth was above a 70% threshold value of the theoretical maximum memory bandwidth for the platform.

  Some applications can execute in phases that use memory bandwidth in a non-uniform manner. For example, an application that has an initialization phase may use more memory bandwidth initially. Use these metrics to identify how the application uses memory through the duration of execution.

NUMA: % of Remote Accesses
: In non-uniform memory architecture (NUMA) machines, memory requests missing last level cache may be serviced either by local or remote DRAM. Memory requests to remote DRAM incur much greater latencies than those to local DRAM. It is recommended to keep as much frequently accessed data local as possible. This metric indicates the percentage of remote accesses, the lower the better.

Vectorization
: The percentage of packed (vectorized) floating point operations. The higher the value, the bigger the vectorized portion of the code. This metric does not account for the actual vector length used for executing vector instructions. As a result, if the code is fully vectorized, but uses a legacy instruction set that only utilizes a half of the vector length, the Vectorization metric is still equal to 100%.

Instruction Mix
: This section contains the breakdown of micro-operations by single precision (SP FLOPs) and double precision (DP FLOPs) floating point and non-floating point (non-FP) operations. SP and DP FLOPs contain next level metrics that enable you to estimate the fractions of packed and scalar operations. Packed operations can be analyzed by the vector length (128, 256, 512-bit) used in the application.
  - **FP Arith/Mem Rd Instr. Ratio:** This metric represents the ratio between arithmetic floating point instructions and memory read instructions. A value less than 0.5 might indicate unaligned data access for vector operations, which can negatively impact the performance of vector instruction execution.
  - **FP Arith/Mem Wr Instr. Ratio:** This metric represents the ratio between arithmetic floating point instructions and memory write instructions. A value less than 0.5 might indicate unaligned data access for vector operations, which can negatively impact the performance of vector instruction execution. The metric value might indicate unaligned access to data for vector operations.

Additional Resources

- Getting Started with Application Performance Snapshot - Linux OS
- Application Performance Snapshot User's Guide for LinuxOS

Article ID: 563
Last updated: 08 Jun, 2022
Revision: 82

Optimizing/Troubleshooting -> Code Development -> Performance Analysis -> Getting a Quick Performance Overview with Intel APS
https://www.nas.nasa.gov/hecc/support/kb/entry/563/